Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 2 of 2 returned.

☐ 1. Document ID: US 20050076007 A1

L4: Entry 1 of 2 File: PGPB Apr 7, 2005

PGPUB-DOCUMENT-NUMBER: 20050076007
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050076007 A1

TITLE: Structured query language table merging

PUBLICATION-DATE: April 7, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Evans, Christopher	Bristol		GB	
Fragapane, Paolo	Bristol		GB	
Cave, Stephen	Gloucester		GB	
Osborn, Andrew	Bristol		GB	

US-CL-CURRENT: 707/2

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

☐ 2. Document ID: US 5963933 A

L4: Entry 2 of 2 File: USPT Oct 5, 1999

US-PAT-NO: 5963933
DOCUMENT-IDENTIFIER: US 5963933 A

TITLE: Efficient implementation of full outer join and anti-join

DATE-ISSUED: October 5, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cheng; Josephine M.	San Jose	CA		
Leung; Ting Yu	San Jose	CA		
Pirahesh; Mir Hamid	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE

International Business Machines Corporation | Armonk NY | 02

APPL-NO: 08/ 882027 [PALM]
DATE FILED: June 25, 1997

INT-CL: [06] G06 F 17/30

US-CL-ISSUED: 707/2; 707/3, 707/4, 707/5, 707/100, 707/102, 707/103
US-CL-CURRENT: 707/2; 707/100, 707/102, 707/3, 707/4, 707/5

FIELD-OF-SEARCH: 707/2-5, 707/100, 707/102, 707/103, 434/336, 434/350, 340/146.2, 395/500, 457/21

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5412804	May 1995	Krishna	707/2
5557791	September 1996	Cheng et al.	395/600
5701454	December 1997	Bhargava	707/2
5765159	June 1998	Srinivasan	707/102
5797136	August 1998	Boyer	707/2

OTHER PUBLICATIONS

"Hypergraph Based Reordering of Outer Join Queries With Compelx Predications", Gautam Bhargava, Piyush Goel & Bala Iyr, ACM 0-89791-731-6/95/0005, 1995, pp. 304-315.
"SQL Query Optimization: Reordering for a General Class of Queries", Piyush Goel & Bala Iyer, ACM 0-89791-794-4/96/0006, 1996, pp. 47-56.

ART-UNIT: 271

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Mizrahi; Diane D.

ATTY-AGENT-FIRM: Gray Cary Ware Freidenrich

ABSTRACT:

Method for specifying SQL "full outer joins" which enables the use of arbitrary join conditions in specifying the query. This is enabled by equating a full outer join with a left outer join unioned with a right outer join less the matched tuples from the right outer join. A number of embodiments further present improvements in execution speed. One such improvement utilizes as a second operand of the union query a novel operator, the "ANTI-JOIN". The anti-join is implemented as a right outer join coupled with an ISNULL predicate.

52 Claims, 7 Drawing figures

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw D

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Term	Documents
FOREIGN	607404
FOREIGNS	15
KEY	809732
KEYS	231084
(3 AND (FOREIGN NEAR KEY)).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	2
(L3 AND (FOREIGN NEAR KEY)).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	2

Display Format: - Change Format

Previous Page Next Page Go to Doc#

L6: Entry 2 of 2 File: USPT Oct 5, 1999

DOCUMENT-IDENTIFIER: US 5963933 A
TITLE: Efficient implementation of full outer join and anti-join

Abstract Text (1):
Method for specifying SQL "full outer joins" which enables the use of arbitrary join conditions in specifying the query. This is enabled by equating a full outer join with a left outer join unioned with a right outer join less the matched tuples from the right outer join. A number of embodiments further present improvements in execution speed. One such improvement utilizes as a second operand of the union query a novel operator, the "ANTI-JOIN". The anti-join is implemented as a right outer join coupled with an ISNULL predicate.

Brief Summary Text (3):
This invention relates to query mechanisms for relational database management systems. More particularly, the present invention relates to a novel method of evaluating SQL "full outer joins", which methodology enables the use of arbitrary join conditions in specifying the query.

Brief Summary Text (5):
A relational database management system, or RDBMS, is a computer implemented database management system that uses relational techniques for storing and retrieving data. Relational databases are computerized information storage and retrieval systems in which data in the form of tables, formally denominated as "relations", are typically stored for use on disk drives or similar mass data stores. A "table" includes a set of rows, formally denominated "tuples" or "records", spanning several columns. Each column in a table includes "restrictions" on the data contents thereof and may be designated as a primary or foreign key. Reference is made to C. J. Date, An Introduction to Database Systems, 6th edition, Addison-Wesley Publishing Co. Reading, Mass. (1994) for a comprehensive general treatment of the relational database art.

Brief Summary Text (14):
This query returns all the matching pairs of T1 and T2 tuples based on the join condition "q1.c1>=q2.c2", just as in a regular join. The query also returns rows of T1 which do not match any T2 rows based on the join condition. In this example, T1 is called the tuple-preserving operand, while T2 is called the null-producing operand, because T1 tuples will be preserved when there is no match in T2.

Brief Summary Text (19):
This query returns all the matching pairs of T1 and T2 tuples based on the join condition "q1.c1>=q2.c2". The query also returns rows of T1 which do not match any T2 rows based on the join condition as well as rows of T2 which do not match any T1 rows.

Brief Summary Text (22):
Many database management systems support left and right outer joins. Some systems further support full outer joins, but they impose the strict limitation that only the conjunction of equality predicates is allowed as the join condition. This

limitation is due to the fact that these systems implement the full outer join using a modified merge join algorithm which, prior to the making of the present invention, has inherently been restricted to the use of equality predicates.

Brief Summary Text (24):
In general, each of these references regarding outer join implementation shares common theme: the strategies taught therein require special runtime operators, which somehow keep track of which tuples must be preserved. In the prior art this has typically been done by creating and maintaining some list of tuple identifiers, or TIDs, for those tuples which must be preserved. When the outer join condition involves only equality predicates in the conjunctive normal form, it is possible to implement the full outer join via a modified merge join algorithm without keeping lists of preserved tuple identifiers. However, a modified merge join, according the prior art, cannot be used for arbitrary join conditions, including but not necessarily limited to the inequality predicate and disjunction, as specified in IPSO-ANSI92. Moreover, the creation and maintenance of TID lists introduces an unwanted and now unnecessary element of computational overhead, thereby inducing system performance problems in any system which relies on such lists.

Brief Summary Text (25):
Join conditions are well defined in IPSO-ANSI92. Typically, join conditions defining the relationship between tuples are logically selectable from the following set of predicates:

Brief Summary Text (28):
equals (=);

Brief Summary Text (29):
does not equal (.noteq.)

Brief Summary Text (32):
is greater than or equal to (.gtoreq.); and

Brief Summary Text (33):
is less than or equal to (.ltoreq.).

Brief Summary Text (34):
A plurality of these conjunction predicates can also be logically joined by the logical operators AND, OR, and NOT. Accordingly, as used herein, the term "arbitrary join conditions" defines any of these join conditions, or a plurality of join conditions joined by one of the defined logical operators. This precisely defined term defines over prior art full outer join methodologies, which support only the equality predicate.

Brief Summary Text (35):
Finally, the computation of large queries implementing a full outer join can be resource intensive. This problem is often exacerbated by increasingly complex query structures where the union of a plurality of outer joins repeatedly returns a number of instances of a matching pair. Any solution to the problem of using a modified merge join to implement a full outer join and which further enables the use of arbitrary join conditions should, to the greatest extent possible, minimize the impact of the query on system resources by only returning the first instance of a matching pair.

Brief Summary Text (36):
Accordingly, there is a clearly-felt need in the art for a methodology which enables the use of a modified merge join to implement a full outer join which enables the use of arbitrary join conditions other than the equality predicate. There is a further need that this query methodology return only a first instance of a matching pair in response to a join condition, thereby enabling the methodology

to be resource efficient. These unresolved problems are clearly felt in the art and are solved by this invention in the novel manner described below.

Brief Summary Text (38):
This invention devises, for the first time, a mechanism which enables the implementation of the full outer join with any join condition, including but not necessarily limited to the inequality predicate and disjunction. This is enabled in one embodiment of the present invention by equating, in a novel manner, a full outer join with a left outer join unioned with a right outer join less the matched tuples from the right outer join. To implement this embodiment, a full outer join query is programmatically rewritten as a union of a left outer join and a select query with a "not-exists" subquery. Indeed, the present invention provides, for the first time, a methodology for properly unioning a left outer join and a right outer join whereby the result is a properly formed full outer join producing a correct answer set.

Brief Summary Text (42):
It is emphasized that the principles of the present invention achieve the several features and advantages hereof without any restriction on the join condition for the full outer join. This is true because there is no restriction on the join condition for left and right outer joins.

Detailed Description Text (15):
A first preferred embodiment of the present invention devises, for the first time, a mechanism which enables a query implementing a full outer join with any join conditions, including but not necessarily limited to the inequality predicate and disjunction. This is enabled by equating, in a novel manner, a full outer join with a left outer join unioned with a right outer join less the matched tuples from the right outer join. Referring to FIG. 4, a flow diagram of this embodiment is shown. At 401, the routine is started, and at 402, the outer join to be performed is identified. At 403, the full outer join is programmatically transformed into a union of a left outer join and a right outer join, minus the matched tuples from the right outer join. The rewritten SQL statement which implements the transformed full outer join is provided to the RDBMS' optimizer (not shown) at 404, and program execution continues at 405. Consider a full outer join with join condition P(T1, T2):

Detailed Description Text (18):
It is recognized that transforming a full outer join into a union query with a not-exists subquery may not provide optimally efficient computational performance. This is due to the fact that many subqueries are inherently slow to execute, especially in the MPP shared-nothing environment. Accordingly, the second preferred embodiment of the present invention presents a novel and computationally efficient full outer join with an arbitrary join condition capability.

Detailed Description Text (19):
Having reference to FIG. 5, this embodiment of the present invention also transforms the full join query into a union query of left outer join and right outer join, at 501-503. However, in order to ensure that the query returns the correct answer, for the right outer join all the matched rows are filtered out from 504-506. To indicate that the output from the right outer join is a matched row, a non-nullable column of the null-producing operand is added to the output of the right outer join at 504. The non-nullable column includes a primary key (key), and a row identifier (rid) or tuple identifier (tid). The value of these columns is set to null at 505 when there is not a match on the tuple-preserving operand by the definition of the outer join. By applying the "IS NULL" predicate after the right outer join at 506, all matched rows are removed from the answer set. The rewritten SQL statement which implements the transformed full outer join is provided to the RDBMS' optimizer (not shown) at 507, and program execution continues at 508.

Detailed Description Text (22):
The preceding approach is acceptable given a non-nullable column. FIG. 6 details another preferred embodiment useful where an outer join operand does not have any non-nulable column, for instance when the operand is a derived table, a column of constants could be added to achieve the same effect. For example, suppose both DT1 and DT2 are not base tables: i.e., their row identifiers were not readily avialable, and all columns were nullable. A full outer join for such a condition could then be rewritten as:

Detailed Description Text (24):
Having continued reference to FIG. 6, this embodiment of the present invention also transforms the full join query into a union query of left outer join and right join, at 601-603. However, in order to ensure that the query returns the correct answer, for the right outer join all the matched rows are filtered out from 604-606. By adding a column of constant 1 at 604, or any other constant, in the null-producing operand of the right outer join, it is assured that the column value is null for the preserved tuples of DT2 (q4) at 605. By applying the "IS NULL" predicate after the right outer join at 606, all matched rows are removed from the answer set. The rewritten SQL statement which implements the transformed full outer join is provided to the RDBMS' optimizer (not shown) at 607, and program execution continues at 608. Hence, those DT2 rows that do not have any matching rows in DT1 are always selectable. This then discloses the third preferred embodiment of the present invention.

Detailed Description Text (27):
Given tables T1 and T2, and ajoin condition P(T1, T2), the anti-join (T1, T2, P(T1, T2)) returns all T1 tuples that do not match with T2 tuples using join condition P (T1, T2).

Detailed Description Text (28):
Having continued reference to FIG. 7, in this embodiment after the full outer join is identified at 702, it is transformed as the union of a left outer join and an anti-join at 703. The rewritten SQL statement which implements the transformed full outer join is provided to the RDBMS' optimizer (not shown) at 704, and program execution continues at 705.

Detailed Description Text (34):
To implement a full outer join using anti-join, consider a full outer join discussed above with join condition P(T1, T2):

Detailed Description Text (37):
It should be emphasized that because the approach taught herein for implementing a full outer join utilizes the properties of the left/right outer join, there is no restriction on the join condition for the full outer join because there is no restriction on the join condition for the left/right outer join which enables it.

CLAIMS:

1. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having one or more columns, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor implemented method for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set requiring the matching of tuples from the first table and the second table, the method comprising the step of joining the tuples of the first and second tables using a full outer join with a join condition, the join condition being free of restrictions as to the type of join condition specified, the step of joining including unioning a left outer join with a right outer join

less the matched tuples from the right outer join.

2. The method of claim 1 wherein the join condition is from the set of comparison operators consisting of: IS NULL; LIKE; EQUALS; DOES NOT EQUAL; IS GREATER THAN; IS LESS THAN; IS GREATER THAN OR EQUAL TO; and IS LESS THAN OR EQUAL TO.

4. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor implemented method for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set requiring the matching of tuples from the first table and the second table, the method comprising the step of joining the tuples of the first and second tables using a full outer join with any join condition specifiable in the query language, the step of joining including unioning a left outer join with a right outer join less the matched tuples from the right outer join.

5. The method of claim 4 wherein the join condition is from the set of comparison operators consisting of: IS NULL; LIKE; EQUALS; DOES NOT EQUAL; IS GREATER THAN; IS LESS THAN; IS GREATER THAN OR EQUAL TO; and IS LESS THAN OR EQUAL TO.

7. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor implemented method for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set requiring the matching of tuples from the first table and the second table, the method comprising the steps of:

for a full outer join condition, receiving a full outer join query of the tuples of the first operand and the tuples of the second operand, with a join condition, wherein the second operand has at least one non-nullable column;

programmatically transforming the full outer join query into a union query of a left outer join of the first and second operands, and a right outer join of the first and second operands; and

filtering from the answer set all the matched rows of the right outer join.

10. The method of claim 9 wherein the step of adding a non-nullable column further comprises the step of including in the non-nullable column at least one of: a primary key; a row identifier; and a tuple identifier.

11. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, both tables being tables other than base tables, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor implemented method for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set requiring the matching of tuples from the first table and the second table, the method comprising the steps of:

for a full outer join condition, receiving a full outer join query of the tuples of

the first operand and the tuples of the second operand, with a join condition, wherein the second operand has the property of being free of non-nullable columns;

programmatically transforming the full outer join query into a union query of a left outer join of the first and second operands and a right outer join of the first and second operands; and

filtering from the answer set all the matched rows of the right outer join.

15. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor implemented method for returning an answer set from the data in the tables responsive to a query to the database management system, the method comprising the steps of:

identifying the query as a full outer join query having an answer set including tuplesles of the first table that do not match tuples of the second table; and

executing an anti-join query of the first and second tables with a join condition using a run time operator with early out property.

16. The method of claim 15 wherein the join condition is from the set of comparison operators consisting of: IS NULL; LIKE; EQUAL; DOES NOT EQUAL; IS GREATER THAN; IS LESS THAN; IS GREATER THAN OR EQUAL TO; and IS LESS THAN OR EQUAL TO.

18. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor implemented method for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set comprising of the first table tuples that do not match the second table tuples, the method comprising the steps of:

receiving an anti-join query of the first and second tables with a join condition, wherein the second table has at least one non-nullable column;

programmatically transforming the anti-join into a left outer join of the first and second tables with an early out property; and

filtering from the answer set all the matched rows of the left outer join.

20. The method of claim 19 wherein the step of adding a non-nullable column further comprises the step of including in the non-nullable column at least one of: a primary key; a row identifier; and a tuple identifier.

21. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor implemented method for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set comprising tuples of the first table that do not match the tuples of the second table, the method comprising the steps of:

receiving an anti-join query of the first and second tables with a join condition, wherein the second table has the property of being free of non-nullable columns;

programmatically transforming the anti-join into a left outer join with an early out property; and

filtering from the answer set all the matched rows of the left outer join.

24. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor implemented method for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set requiring the matching of tuples from the first table and the second table, the method comprising the steps of:

receiving a full outer-join query of the tuples of the first tables with a join condition;

programmatically transforming the full outer join query into a union query of a left outer join of the first and second operands, and an anti-join of the first and second operands using a new run time operator.

25. The method of claim 24 wherein the join condition is from the set of comparison operators consisting of: IS NULL; LIKE; EQUALS; DOES NOT EQUAL; IS GREATER THAN; IS LESS THAN; IS GREATER THAN OR EQUAL TO; and IS LESS THAN OR EQUAL TO.

27. A computer-implemented apparatus, with bus means, processor means, data storage means, input and output means and a relational database management system, for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having one or more columns, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, which apparatus returns an answer set from the data in the tables responsive to a query to the database management system, the apparatus comprising programmatic means for identifying the query as a full outer join query having an answer set requiring the matching of tuples from the first table and the second table, and programmatic means for joining the tuples of the first and second tables using a full outer join with a join condition, the join condition having the property of being free of restrictions as to the type of join condition specified, the means for joining comprising programmatic means for unioning a left outer join with a right outer join less the matched tuples from the right outer join.

28. The apparatus of claim 27 wherein the join condition is from the set of comparison operators consisting of: IS NULL; LIKE; EQUALS; DOES NOT EQUAL; IS GREATER THAN; IS LESS THAN; IS GREATER THAN OR EQUAL TO; and IS LESS THAN OR EQUAL TO.

30. A computer-implemented apparatus, with bus means, processor means, data storage means, input and output means and a relational database management system, for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, the apparatus for returning an answer set from the data in the tables responsive to a query to the database management system, the apparatus comprising programmatic means for identifying the query as a full outer join query having an answer set requiring the matching of tuples from the first

table and the second table, and programmatic means for joining the tuples of the first and second tables using a full outer join with any join condition specifiable in the query language, the means for joining comprising means for unioning a left outer join with a right outer join less the matched tuples from the right outer join.

31. The apparatus of claim 30 wherein the join condition is from the set of comparison operators consisting of: IS NULL; LIKE; EQUALS; DOES NOT EQUAL; IS GREATER THAN; IS LESS THAN; IS GREATER THAN OR EQUAL TO; and IS LESS THAN OR EQUAL TO.

33. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processing apparatus ("apparatus") for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set requiring the matching of tuples from the first table and the second table, the apparatus comprising:

for a full outer join condition, programmatic means for receiving a full outer join query of the tuples of the first operand and the tuples of the second operand, with a join condition, wherein the second operand has at least one non-nullable column;

programmatic means for transforming the full outer join query into a union query of a left outer join of the first and second operands and a right outer join of the first and second operands; and

programmatic means for filtering from the answer set all the matched rows of the right outer join.

36. The apparatus of claim 35 wherein the programmatic means for adding a non-nullable column further comprises programmatic means for including in the non-nullable column at least one of: a primary key; a row identifier; and a tuple identifier.

37. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processing apparatus ("apparatus") for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set requiring the matching of tuples from the first table and the second table, the apparatus comprising:

for a full outer join condition, programmatic means for receiving a full outer join query of the tuples of the first operand and the tuples of the second operand, with a join condition, wherein the second operand has the property of being free of non-nullable columns;

programmatic means for transforming the full outer join query into a union query of a left outer join of the first and second operands and a right outer join of the first and second operands; and

programmatic means for filtering from the answer set all the matched rows of the right outer join.

41. In a relational database management system utilizing a data processor for

storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor for returning an answer set form the data in the tables responsive to a query to the database management system, the data processor comprising:

means for identifying the query as a full outer join query having an answer set including tuples of the first table that do not match tuples of the second table; and

means for executing an anti-join query of the first and second tables with a join condition using a new run time operator with early out property.

42. The data processor of claim 41 wherein the join condition is from the set of comparison operators consisting of: IS NULL; LIKE; EQUAL; DOES NOT EQUAL; IS GREATER THAN; IS LESS THAN; IS GREATER THAN OR EQUAL TO; and IS LESS THAN OR EQUAL TO.

44. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set comprising of the first table tuples that do not match the second table tuples, the data processor comprising:

means for receiving an anti-join query of the first and second tables with a join condition, wherein the second table has at least one non-nullable column;

means for programmatically transforming the anti-join into a left outer join of the first and second tables with an early out property; and

means for filtering from the answer set all the matched rows of the left outer join.

46. The data processor of claim 45 wherein the function of adding a non-nullable column further comprises the function of including in the non-nullable column at least one of: a primary key; a row identifier; and a tuple identifier.

47. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set comprising of the first table tuples that do not match the second table tuples, the data processor comprising:

means for receiving an anti-join query of the first and second tables with a join condition, wherein the second table has the property of being free of non-nullable columns;

means for programmatically transforming the anti-join into a left outer join with an early out property; and

means for filtering from the answer set all the matched rows of the left outer join.

50. In a relational database management system utilizing a data processor for storing data in the form of at least a first table defining a first operand and a second table defining a second operand, the first and second tables comprised of tuples having at least one column, wherein the data contained in the relational database management system is retrievable by means of query language queries to the database management system, a data processor for returning an answer set from the data in the tables responsive to a query to the database management system, the answer set requiring the matching of tuples from the first table and the second table, the data processor comprising:

means for receiving a full outer-join query of the tuples of the first tables with a join condition; and

means for programmatically transforming the full outer join query into a union query of a left outer join of the first and second operands, and an anti-join of the first and second operands using a run time operator.

51. The data processor of claim 50 wherein the join condition is from the set of comparison operators consisting of: IS NULL; LIKE; EQUALS; DOES NOT EQUAL; IS GREATER THAN; IS LESS THAN; IS GREATER THAN OR EQUAL TO; and IS LESS THAN OR EQUAL TO.

Previous Doc Next Doc Go to Doc#